

# Implementing a Dialog System for Restaurant Recommendations

DIEGO FARINELLA, Utrecht University, The Netherlands

THIJS VISSER, Utrecht University, The Netherlands

DEEN WILLEMSSEN, Utrecht University, The Netherlands

ALEX DE ROODE, Utrecht University, Netherlands

Additional Key Words and Phrases: Dialogue System, Machine Learning, Restaurant Recommendation

*Code and data:* <https://github.com/DeenWillemssen/methodsaipart1>

**Abstract** The objective of the present work is to present the development of a dialog system tailored for restaurant recommendations. The system's task can be briefly described as interacting with the user by means of a text-only conversation and ultimately suggesting to the user the most suitable restaurant for their requirements among the ones in the system's dataset. In order to do so, we first built multiple machine learning models in order to label the user's objective in each utterance. Then, exploiting machine learning results, we constructed a state transition diagram and accordingly a state transition function that guides the system in its interactions: the dialog system is to be conceptualized as an agent that switches its state according to the user's acts and performs different actions in each state. Additionally, the system has been provided with four different configurability options to allow users to personalize their interactions. The system developed here is able to store a limited amount of user preferences, and of a limited complexity: it is not able to grasp food preferences that require more words to be expressed (e.g. 'central european'), or to differentiate between price ranges in a continuous way (it only recognizes the attributes 'cheap', 'moderate', 'expensive'). Therefore, among the possible improvements for our dialog system, we suggest the system to be allowed to grasp more complex pattern in the users' preferences. Also the development of more a precise classifier for user's speech act is to be suggested among possible improvements.

## 1 INTRODUCTION

In this report, we present the development of a dialog system tailored for restaurant recommendation. Our dialog system interacts with users by means of text-only conversations and suggests to the users the most suitable restaurant according to their wishes. In order to develop the final system, we refer to the Dialog State Tracking Challenge (DSTC 2) dataset, presented in Section 2: a collection of system-user interactions in the domain of restaurant recommendations. The system built in this project has three core components.

The first core component of the final system is a machine learning model trained on the dataset, which aims to label users' utterances according to their objective. Section 3 presents the development of multiple models of this sort and compares their performances in order to select the most suitable to be used in the final system. The second component of our dialog system consists of a state transition function that guides the system in its interaction with the user and prescribes different actions to the system according to the flux of the interaction. In Section 4 we present such function based on a state transition diagram constructed in Section 2.2. Additionally, we present examples of system-user interactions and discuss how the state transition function manages the flux of the dialog. The third core component of the system is a reasoning component. This component, based on certain specific properties of the restaurant, infers the boolean values of other attributes and also explains to the user the inference, and thus the reason, for a restaurant to be assigned to a specific attribute. This reasoning component of the system is presented in Section 5.

---

Authors' addresses: Diego Farinella, [d.farinella@students.uu.nl](mailto:d.farinella@students.uu.nl), Utrecht University, Student Number: 7675763, Utrecht, The Netherlands; Thijs Visser, [t.visser@students10@students.uu.nl](mailto:t.visser@students10@students.uu.nl), Utrecht University, Student Number: 9787941, Utrecht, The Netherlands; Deen Willemsen, [d.p.willemsen@students.uu.nl](mailto:d.p.willemsen@students.uu.nl), Utrecht University, Student Number: 5306630, Utrecht, The Netherlands; Alex de Roode, [a.deroode@students.uu.nl](mailto:a.deroode@students.uu.nl), Utrecht University, Student Number: 6779484, Utrecht, Netherlands.

Ultimately, in Section 6, additional configurability options provided to the system are presented and discussed in their implementation. We then conclude in Section 7, casting a final eye on the whole project and identifying pitfalls and possible improvements for the future.

## 2 DATA

This section aims at presenting the dataset employed for the development of the dialog system and discuss in detail his properties considered relevant for the development of the system. Subsequently, (2.2 the dialog state transition diagram will be presented, as well as some dialog snippets to better highlight the functioning of it.

### 2.1 Dataset analysis

The Dialog State Tracking Challenge (DSTC 2)<sup>1</sup> consists in 3,235 dialogs in English between a restaurant recommendation system and a user. More specifically, in each dialog instance the user is given a specific task to obtain in the domain of restaurant recommendations and interacts with the system to achieve it. An example of a task given to the user is: «You want to find an expensive restaurant and it should serve european food. You want to know the address and phone number of the venue». <sup>2</sup> Each turn in the dialog (namely, each prompt of the user) is labeled among 15 classes or dialog acts that indicates the objective of the user in that utterance. The 15 dialog act classes are: 'ack' (acknowledgment), 'affirm' (confirming), 'bye' (ending the conversation), 'confirm' (checking a given information), 'deny' (disagree), 'hello' (start the dialog), 'inform' (indicate a preference), 'negate' (negation), 'null' (not meaningful interaction), 'repeat' (ask a repetition), 'reqalts' (asking for alternatives), 'reqmore' (asking for more suggestions), 'request' (ask for an information), 'restart' (restarting the interaction), 'thankyou' (thanking).

Two relevant features of the database need to be highlighted. The former is that the dialogues present in the dataset are collected by means of automatic speech recognition (ASR): this is relevant since it influences the quality of the dialog in terms of transcription and also affects the tone of the user as regards the formality level of the interaction. In particular, the informal tone of the dataset might affect the classification of user utterances.<sup>3</sup> The latter relevant feature of the dataset is the large number of duplicate utterances. Largely repeated sentences, namely a lower data variety, might be overweighted by the system and thus lead to overfitting.

### 2.2 State Transition Diagram

Based on the dataset provided, a state transition diagram has been constructed to guide the dialog system in the interaction with the users. In this section, we briefly describe the diagram and then provide some dialog snippets to be compared with it in order to provide a precise understanding of how the transitions between different states work.

In short, the state transition diagram constructed (Figure 1) has a total of 11 states, each representing a different situation in the process of interacting with a user in which the system has different kinds of information. More specifically, the states in the diagram are as follows:

- State 1 - Hello: starting the interaction with the user;
- State 2 - Check Preferences: Storing the preferences of the user;
- State 3 - Preference Loop: ask each preference until all are collected;
- State 4 - Match Preferences to Restaurants: retrieve the matching restaurants(s) from the dataset;

<sup>1</sup>The dataset can be found at <https://www.microsoft.com/en-us/research/event/dialog-state-tracking-challenge>

<sup>2</sup>Task 02241.

<sup>3</sup>This aspect becomes particularly evident during the development of a baseline system based on keywords (See Section 3.1).

- State 5 - No Match: no restaurant suitable is found;
- State 6 - Match Success: when a restaurant suiting the user wishes is found;
- State 7 - Requirement Loop: storing the extra requirements of the user;
- State 8 - User Informed: the user is informed of the found match;
- State 9 - No Extra Requirements Matches Left: when no match is left according to the user's extra requirements;
- State 10 - Fetch Data: asking the user which data are to be collected from the found match;
- State 11 - No Data: the data asked is not present;

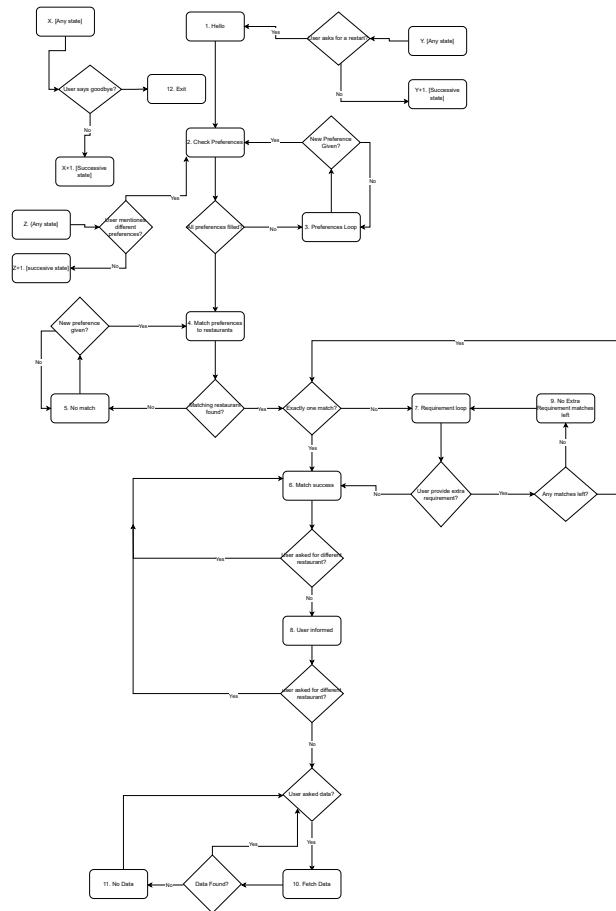


Fig. 1. State Transition Diagram

At this point, some dialog snippets found in the dataset are helpful to better understand the logic of the state transition diagram. Let's analyze a first system-user interaction (Dialog 1).<sup>4</sup> At the beginning of this instance, the system is in state

<sup>4</sup>This full dialog can be found in the original dataset at session id: *voip - df8b50918 - 20130327\_024718*

8 (User Informed) and is providing a suitable restaurant to the user. Subsequently, the user asks for more information first about the food and then about the address of the restaurant: accordingly, the system enters in state 10 (Fetch Data) and retrieves the correct information to share with the user. Another interesting example can be Observed in Dialog 2.<sup>5</sup> In this case, the system starts in state 1 (Hello) and directly passes in state 2 (Check Preferences). Then, due to a not meaningful utterance of the user, the system has no preferences stored, thus it enters in state 3 (Preference Loop) and asks the user all the missing preferences one by one. Once all preferences have been collected, the system switches to states 4 (Match Preferences to Restaurants) and retrieves a restaurant, entering state 6 (Match Success) and consequently 8 (User Informed), informing the user of the found match.

#### Dialog 1

**system:** "the missing sock is a nice place in the east of town and the prices are cheap"

**user:** "what type of food"

**system:** "the missing sock serves international food"

**user:** "what is the address"

**system:** "Sure, the missing sock is on Finders Corner Newmarket Road"

**user:** "thank you good bye"

#### Dialog 2

**system:** "Hello, welcome to the Cambridge restaurant system? ..."

**user:** "noise"

**system:** "What part of town do you have in mind?"

**user:** "the west part of town id like a cheap restaurant"

**system:** "What kind of food would you like?"

**user:** "any kind of food"

**system:** "la margherita is a nice place in the west of town and the prices are cheap"

### 3 MACHINE LEARNING

In this section, an overview of the implementation of four machine learning models labeling users' utterances is provided. However, before delving into detail regarding the machine learning model constructed, it is indeed necessary to cast an eye on two steps undertaken: the process of data preprocessing and the development of two baseline systems.

#### 3.1 Baselines

Before constructing the baselines, some very simple preprocessing was applied to the data: namely, tokenisation of the utterances as well as vectorisation (creating BoW representations of the utterances), but also cleaning of the data by lower-casing all inputs of the dataset. The first baseline simply consists in assigning each instance to the most common label in the dataset. A very basic statistical analysis on the data revealed that the most widespread label in the dataset is 'inform'<sup>6</sup>. The accuracy of the first baseline is just 40%, which is expected given the very naive approach that this benchmark represents. The second baseline system employs a slightly more sophisticated approach, though still a basic one. The approach consists in assigning an utterance to a label based on the occurrence of some keywords used for each dialog act: e.g., 'i dont', 'no', 'not', 'wrong', 'dont want' are the key words for the label 'deny'. The selection of keywords was simply based on the most frequent words for each label and, given the very informal tone of the dataset<sup>7</sup>

<sup>5</sup>This full dialog corresponds to session id: *voip - da10d74c3e - 20130326\_001047* in the original dataset

<sup>6</sup>Followed by: request, thankyou, and reqalts.

<sup>7</sup>It has already been pointed out that the dataset has been produced by means of ASR, which significantly affect the tone of the user and makes it likely to have non-grammatical sentences and words (See Section 2.1).

the list of keywords also includes non-grammatical words and abbreviations. This second baseline system considerably improves performance and achieves an accuracy of 85%, even reaching perfect precision with respect to three labels.<sup>8</sup>

### 3.2 Machine Learning Models

Having discussed the development and results of the two baseline systems, four machine learning models for predicting the correct dialog act given a user’s input are now presented. Before developing the models, a statistical analysis of the data in order to set the best parameters for the machine learning systems was performed. This brief analysis revealed two core points. The first one is related to the fact that utterance lengths are varied in the dataset, but the average length is quite high (around 19 characters). This indicates that far more utterances are more likely to carry or ask information than to just affirm or negate; in fact, asking or communicating information requires more words. The second core point is related to label distribution in the dataset: speech acts are not equally distributed across the dataset. The most common labels are inform 39%, request 25% and thankyou 12%; while classes such as confirm (0.6%) or affirm (4%) are significantly less represented. These two points have been taken into account when setting the parameters of the models. Furthermore, each model has been trained on two different versions of the dataset: the original one, and a deduplicated version of it<sup>9</sup>. The original dataset has been splitted in an 85% training set and a 15% test set. A second train-test split (also 85%-15%) was performed after removing repeated utterances from the original dataset. The different version of the dataset used for training drastically affects the performance of the model. In the following, we present each model and its parameters, and ultimately we perform an error analysis and a comparison between the results achieved by the models.

The first model consists of a very simple decision tree with the following parameters: Gini impurity as criterion for split<sup>10</sup>; deterministical split; no maximum depth. As a second machine learning system, we implemented a random forest classifier with the same parameters used for the decision tree. The third system developed is a logistic regression classifier. Ultimately, we implemented a feedforward neural network (multi-layer perceptron) classifier with adaptive moment estimation (adam). The network was implemented with two hidden layers respectively of size 64 and 32 units. For both hidden layers the activation function used is the default ReLu, and for the output layer the model uses Softmax for multi-class problems as this task.<sup>11</sup> The performances of each model, both in the version trained on the original dataset and the one trained on the deduplicated one, are reported in Table 1.

Table 1. Performance metrics of machine learning models on original and deduplicated datasets

Model	Dataset	Accuracy (%)	Log Loss
Decision Tree	Original	97	0.923
	Deduplicated	89	4.079
Random Forest	Original	98	0.130
	Deduplicated	89	0.440
Logistic Reg.	Original	98	0.085
	Deduplicated	91	0.284
MLP (Adam)	Original	98	0.145
	Deduplicated	92	0.502

<sup>8</sup>More specifically: ‘affirm’, ‘reqmore’ and ‘restart’.

<sup>9</sup>In this version, repeated user utterances have been deleted by the dataset.

<sup>10</sup>The formula of Gini impurity is  $G(t) = \sum_{k=1}^K p_k(1 - p_k) = 1 - \sum_{k=1}^K p_k^2$ , where  $p_k$  is the fraction of class  $k$  in node  $t$ . A detailed analysis of this criterion for decision tree classifiers can be found in Tangirala’s et al. paper [1].

<sup>11</sup>For a detailed survey on multi layer perceptrons with the adam optimizer see Zarafshan’s et al. contribution [2]

### 3.3 Error analysis and models comparison

In general, the labels that appear to be mostly misclassified by the models are 'ack', 'repeat', 'reqmore', 'bye' and 'null'. This is due to their low diffusion in the dataset, both in the original and deduplicated version. Additionally, certain utterances appear to be very difficult to label. As an example, we identified these two utterances: "any thing else", and "um doesnt matter if its in the center of the town".<sup>12</sup> The former utterance is to be classified as 'null', since ASR cause a non-grammatical sentence, the second is to be classified as 'inform', since the user is communicating their indifference to the area of the restaurant.

We ran our models, in the deduplicated version, on these difficult instances and recorded the results (Table 2). While all our models are able to predict the correct label on the second difficult utterance, the logistic regression classifiers performs a wrong prediction on the first utterance.

Table 2. Predictions of different models for difficult utterances

Utterance	DT	RF	LR	MLP
"any thing else"	null	null	inform	null
"um doesnt matter if its in the center of the town"	inform	inform	inform	inform

As a second step of the error analysis, we created three trigger utterances. These latter aim to prove the model against difficult cases and elicit a wrong label. We focused on patterns that are usually difficult for the models to classify, either because they are poorly represented in the dataset, or are non-grammatical sentences produced by ASR. As first triggering sentence, we used the instance "i dont want a chinese restaurant": this should not be easy for the model to classify because the label 'negate' is not widespread in the dataset (1%). The other two utterances produced are: "bye bye north east", and "give me east part of food"; the difficulty in these cases arises from the fact that these sentences are non-grammatical, and hence should be labeled as 'null'. However, their patterns resemble patterns of other labels (such as 'bye' or 'inform'), which would likely trigger the models. The results of this test can be consulted in Table 3.

Table 3. Predictions of different models for triggering utterances

Utterance	DT	RF	LR	MLP
"i dont want a chinese restaurant"	inform	inform	inform	inform
"bye bye north east."	thankyou	bye	bye	bye
"give me east part of food."	inform	inform	inform	request

At this point, we have enough information to perform a confident model comparison. Based on the quantitative performance presented in Section 3.2, we can draw two conclusions. Initially, the models trained on the deduplicated version of the dataset achieve poorer accuracy and a higher logged loss. This is due to the fact that there is no case where a sentence in the training set reappears in the test set. As previously highlighted,<sup>13</sup> this prevents the models to overfit the data and hence guarantees a better classification power of the models. In our case, this means that models trained on the deduplicated version of the dataset are likely to generalize better on unseen data. Now, we shall identify the best performing model. Looking at the quantitative evaluation metrics presented in Section 3.2, it seems pretty clear

<sup>12</sup>These utterances can be found respectively at session id: *voip-03c2655d43-20130327\_194746* and session id: *voip-b20968d1ea-20130323\_112309*.

<sup>13</sup>See Section 2

that the logistic regression classifier achieves the best results: in fact, while the accuracy of the models is pretty balanced (between 89% and 92% for the deduplicated versions), the log loss of the logistic regression model is considerably lower (0.284 against 0.502 of the multi layer perceptron classifier). In conclusion, our analysis indicates that the logistic regression classifier has the most accurate probabilities estimation, and is indeed the most confident model among the ones proposed.

## 4 DIALOG MANAGER

### 4.1 State transition function

In this section, we present the state transition function that guides the system in its interaction with the user. In order to describe such function, the cleanest and most simple approach is to traverse flux of the state transition diagram from the start to the end state, and comment which kinds of checks does the system implement in order to switch its state. Appendix A contains a full dialog example (Dialog 8) that traverses the state transition diagram from the start to the end state.

At the beginning of the interaction the state machine starts in state 1 (Hello) and asks the users preferences. Then, the system switches to state 2 (Check Preferences): here, it checks if all preferences are filled in or not. In case not all preferences are collected<sup>14</sup> the system enters state 3 (Preference Loop), where it asks one remaining preference and go back to state 2 to check once again if all preferences are filled in. This is done until all preferences are collected. In Dialog 3 there is an example of dialog leading the system to state 3.

#### Dialog 3

<b>system:</b>	Hello, welcome to B2's restaurant finder, please inform us of a pricerange, area and foodtype you are interested in
<b>user:</b>	hello i would like an expensive restaurant in the north
<b>system:</b>	Could you tell me your preferred food?

User preferences are collected according to the data present in the restaurant dataset. However, in order to manage likely typing errors committed by the users, the user collects preferences in a more flexible way by recognizing words that have a Levenshtein distance smaller than 3 from values of the restaurants dataset.<sup>15</sup> When all preferences are filled in, the system enters state 4 (Matching Preferences to Restaurants). Here, it retrieves all restaurants that match with the users' preferences. If no matching restaurant is found, the system switches to state 5 (No Match) until different preferences are given, in order to go back to state 4 and check again for restaurants. If only one match is found, the system enters state 6 (Matchsuccess). If there is more than 1 match found, the flux approaches state 7 (requirements loop). Here, the system asks if the user has any extra requirements (e.g., 'romantic','touristic'). If the user has no extra requirements the system returns to state 6 and presents the first restaurant in the list of matches. Otherwise, it filters the restaurants based on the user requirement. If there are no matches for this requirement, the system will enter state 9 (No Extra Requirements Match Left), where the user is asked to choose another requirement, entering state 7 again. In Dialog 4, there is an example of the system going through the extra requirements cycle.

<sup>14</sup>Note that a preference might also be "any", indicating that the user is indifferent to that field but has filled it in

<sup>15</sup>For the sake of precision, we acknowledge that we applied a more strict threshold (1) for words shorter than 5 characters: this avoids errors in storing the preferences.

## Dialog 4

**system:** We currently have found 3 restaurants that comply with your preferences. Would you like to give any additional requirements. We can filter on: child friendliness, romantic and touristic

**user:** i want it to be romantic

**system:** Im sorry but the extra requirements you provided resulted in 0 matches, please provide us with new requirements or do not to get the restaurants we found without them

When the system retrieves a match after the extra requirements are given, it checks if there is only one match or multiples. In case it is only one, it enters state 6 (Matchsuccess) and presents the match, otherwise it goes back to state 7 and checks if there are the user has additional requirements to filter on. In state 6, the system informs the user of the found match and then enters state 8 (User informed of Match). Here, the user can ask for different restaurants if the system has retrieved multiple options: in such case, the system goes back to state 6 and informs the user. In state 8, the user can also ask for more information about the given restaurant, e.g. address or phone number. In such case, the system enters state 10 (Fetch Data): if the data is found, it is communicated and then the system returns to state 8. Otherwise, the system switches to state 11 and informs the user that there is no data available, and ultimately returns to state 8.

From any state, the user can change his preferences: this decision is detected by means of the 'reqalts' speech act. The system will then check what preference the user has changed and go back to state 2 (Check Preferences) and the following ones according to the flux. Dialog 5 presents the case where, after a match, the user changes a preference, and hence the system retrieves restaurants according to the new preference expressed.

## Dialog 5

**system:** I found a nice restaurant called mahal of cambridge serving indian food, at a cheap price in the centre part of town. Any other information about the restaurant we can help you with?

**user:** how about spanish food?

**system:** We currently have found 2 restaurants that comply with your preferences. Would you like to give any additional requirements. We can filter on: child friendliness, romantic and touristic

From any state, if the system detects the speech act 'bye', it enters state 12 (Exit) and the interaction ends. Also, if the speech act 'restart' is detected from any state, the preferences of the user are reset and the system restarts the interaction from state 1.

## 4.2 Response Temeplates

In this Section we briefly present, in Table 4, all the templates of the system's responses to the user, as well as a concrete example for each template.<sup>16</sup>

<sup>16</sup>To be precise, due to the configurability option discussed in Section 6, there are also 'informal' templates that can be turned on by the user. For brevity, we report those templates in Appendix B.



State	Template	Example
HELLO	"Hello, welcome to B2's restaurant finder, please inform us of a pricerange, area and foodtype you are interested in"	"Hello, welcome to B2's restaurant finder, please inform us of a pricerange, area and foodtype you are interested in"
PREFLOOP	"Could you tell me your preferred <slot>?"	"Could you tell me your preferred area?"
NOMATCH	"Deepest apologies but I couldn't find a <pricerange> restaurant serving <food> food in the <area> part of town. Do you have an alternative pricerange, area or food?"	"Deepest apologies but I couldn't find a cheap restaurant serving french food in the center part of town. Do you have an alternative pricerange, area or food?"
MATCHSUCCESS (no extra req.)	"I found a nice restaurant called <name> serving <food> food, at a <pricerange> price in the <area> part of town. Any other information about the restaurant we can help you with?"	"I found a nice restaurant called pizza hut fen ditton serving italian food, at a moerate price in the east part of town. Any other information about the restaurant we can help you with?"
MATCHSUCCESS (with extra req.)	"I found a nice restaurant called <name> serving <food> food, at a <pricerange> price in the <area> part of town.The restaurant is <req> because <reason>. Any other information about the restaurant we can help you with?"	"I found a nice restaurant called mahal of cambridge serving indian food, at a cheap price in the centre part of town.The restaurant is touristic because it has good food and a cheap price. Any other information about the restaurant we can help you with?"
USERMATCHED	"Thank you for using our service, any other information about the restaurant i can help you with?"	"Thank you for using our service, any other information about the restaurant i can help you with?"
FETCHDATA (all requested fields found)	"Sure — the <key 1> is <value 1>; the <key 2> is <value 2>; ..."	"Sure — the phone is 01223 360409; the addr is 3 - 5 millers yard mill lane."
FETCHDATA (some found, some missing)	"Here's what I found: the <key 1> is <value 1>; ... I don't have <missing key 1>; <missing key 2>."	"Here's what I found: the price is cheap; ... I don't have addr."
FETCHDATA (all missing)	"Sorry, I couldn't find the <missing key 1>; <missing key 2> for this restaurant."	"Sorry, I couldn't find the phone number, addr for this restaurant."
FETCHDATA (error)	"Our deepest apologies but something went wrong, we could not retrieve the data you asked for."	"Our deepest apologies but something went wrong, we could not retrieve the data you asked for."
REQLOOP	"We currently have found <n> restaurants that comply with your preferences. Would you like to give any additional requirements. We can filter on: child friendliness, romantic and touristic"	"We currently have found 8 restaurants that comply with your preferences. Would you like to give any additional requirements. We can filter on: child friendliness, romantic and touristic"
NOREQMATCH	"Im sorry but the extra requirements you provided resulted in 0 matches, please provide us with new requirements or do not to get the restaurants we found without them"	"Im sorry but the extra requirements you provided resulted in 0 matches, please provide us with new requirements or do not to get the restaurants we found without them"
EXIT	"Thanks! Goodbye."	"Thanks! Goodbye."

Table 4. System utterance templates and examples

## 5 REASONING

This Section aims at presenting the reasoning component of our dialog system, which is necessary to deal with extra requirements provided by the user. As discussed in Section 4, in case the initial filtering provides more than one match, the system asks the user to provide extra requirements. The reasoning component is used to determine if these extra requirements apply to a restaurant based on logical inference on the properties of the restaurants and then explains to the user why the chosen restaurant applies for the given requirement.

### 5.1 Adding Extra Properties, Filtering and Explaining

So as to make use of the inference rules of the reasoning component ,the following extra properties have been added to the original restaurant dataset:

- food quality: with values *"terrible"*, *"insufficient"*, *"average"*, *"good"*, *"excellent"*;
- crowdedness: with values *"empty"*, *"calm"*, *"normal"*, *"busy"*, *"overcrowded"*;
- length of stay: integer with values between 5 and 180 minutes;

Having added such extra properties to the dataset, the inference rules presented in Table ?? have been applied on the restaurants and saved into the CSV file. The system already saves the results of the rules instead of retrieving them manually later, which is more efficient. When the user presents an extra requirement, the system checks for all matched restaurants if such requirement has value *true* or *false*. In case only one match left, or in case the user chooses not to

look at any other requirements, the system will return the match and also explains the reason why such restaurant complies with the user requirements (namely, presents the inference rule used).

In order to provide explanations the system once again checks if the requirement really applies to the restaurant and if so returns the requirement with the explained reason to the user. If a user provides multiple extra requirements, we assume that the first one was the most important, and hence our system provides the explanation only for the first one. If the user has more than one remaining option and chooses to stop presenting extra requirements, the first match in the queue is presented to the user.

Antecedent	Consequent	t/f
cheap AND good food	touristic	true
Romanian	touristic	false
busy	assigned seats	true
long stay	children	false
busy	romantic	false
long stay	romantic	true

Table 5. Inference rules

## 5.2 Examples of application of the reasoning component

In this brief subsection, we present a few dialog snippets where the reasoning component of our dialog system is involved.

In Dialog 6, we can observe the case of a user providing an extra requirement and then choosing not to filter with more requirements, even tho there is more than one match found; this instance has already been discussed in Section 5.1.

### Dialog 6

**system:** "We currently have found 8 restaurants that comply with your preferences. Do want to give any additional requirements? We can filter on: assigned seating, child friendliness, romantic and toursitic"

**user:** "I want it to be romantic"

**system:** "We currently have found 3 restaurants that comply with your preferences. Do want to give any additional requirements? We can filter on: assigned seating, child friendliness, romantic and toursitic"

**user:** "no"

**system:** "I found a nice restaurant called pizza hut cherry hinton serving italian food, at a moderate price in the south part of town. The restaurant is romantic because people stay for long and it is not busy. Is there anything else?"

Dialog 7 represents an instance where the system does not find any suitable match with the extra requirement provided by the user, and hence communicates that there is no restaurant complying with the provided needs and asks the user fo filter with a different requirement.

## Dialog 7

**system:** "We currently have found 3 restaurants that comply with your preferences. Do want to give any additional requirements? We can filter on: assigned seating, child friendliness, romantic and touristic"

**user:** romantic please

**system:** "I'm sorry but the extra requirements you provided resulted in 0 matches, provide us with new requirements or do not to get the restaurants we found without them"

### 5.3 Contradictions

We point out that inference rules can cause contradictions, namely can cause an attribute of a restaurant (e.g.) 'romantic', to be both *true* and *false*. As an example: according to the inference rules, a restaurant that is *cheap* and serves *good food* is touristic; however, a restaurant that serves *romanian* food is not touristic. Then, in case a cheap romanian restaurant serves good, is it touristic or not? We solved such contradictions by overweighting the *false* values: that is, in case of contradictions the value of the indicated attribute is always *false*.

## 6 CONFIGURABILITY

The system has ultimately be furnished by four different configurability options. Some of these are inherent simply to the graphic experience of the user while interacting with the system, while others directly affect the performance of the dialog system. In this section, we discuss how the configurabilities can be selected by the user, and then ultimately briefly present the four configurability options one by one.

Before starting any interaction, the system informs the user that there are four different configurability options available: 'allcaps', 'delay', 'informal' and 'baseline'. The system also communicates to the user that it is possible to select one of these options, more than one, or no one, and explains how to activate them: the user simply has to type (as their first input) the name of the configurabilities that they wish to activate. In order to provide confirm that the system turned a specific option on, a signal is printed for each option that has been selected by the user. This method allows the user to selected the desired configurability options before starting to interact with the system.

Having clarified how the configurability selection happens, we shall briefly describe each option. The 'allcaps' configurability simply means that the system will print its replies in capital letters. The 'delay' option introduces a delay of three seconds in the output of the system, making the user wait to see the replies. The 'informal' configurability causes the system responses to be less formal and more colloquial. The complete collection of all the 'informal' templates is to be found in Appendix B. Ultimately, the 'baseline' configurability prompts the system to classify the dialog act of the user with the keyword baseline described in section 3.1 instead of relying on the linear regression model. This latter option is the only one that affects the performance of the system itself, while the other options simply relate to the user experience. All the options were implemented in the code by means of simple boolean conditions to be checked either at the moment of printing the system's reply (in the case of 'allcaps', 'delay' and 'informal') or at the classification (in the case of the 'baseline' configurability).

## 7 CONCLUSION

In this report, we discussed the implementation of a dialog system for restaurant recommendation and described the three core components of this system. First of all, a machine learning classifier capable to label user utterances in terms

of speech acts. Then, a dialog manager that guides the system in the interaction with the user according to a state transition diagram. This second core component basically prompts the system to undertake different actions based on the flux of the interaction with the user, and hence represents the heart of our dialog system. Ultimately, a reasoning component of our system infers certain attributes of restaurants based on their properties and communicates such inferences to the user when presenting a match. In addition, we also discussed four different configurability options that have been provided to the system.

As emerges from the implementation of this project, our system is not flawless. Multiple improvements can be added to the actual version of the dialog system in order to optimize its performance. In the remaining of this section, we point out two possible improvements. First of all, the current version of the system is not able to grasp food preferences that span over more than one word (e.g., 'central european'): as a consequence, it is not able to retrieve certain cuisines from the restaurant dataset. As far as now, the system also stores only a few categorical values for attributes such as *price*, while a continuous range of values could be introduced. Additionally, we acknowledge that the speech act classifier employed in the current version of the model lacks accuracy in specific contexts: as an example, the word "restart" is not labeled as a 'restart' speech act. Likely, it is possible to develop a better performing classifier, capable to more accurately identifying patterns in the dataset.

## 8 CONTRIBUTIONS

task	Diego	Thijs	Deen	Alex
<i>data analytics</i>	0.5 hr	—	—	—
<i>keyword baseline</i>	—	—	3 hr	—
<i>BoW representation</i>	—	—	—	3 hr
<i>data deduplication</i>	—	—	—	2 hr
<i>model implementation</i>	2 hr	4.5 hr	—	2 hr
<i>model evaluation</i>	2 hr	—	—	—
<i>error analysis</i>	2 hr	—	—	—
<i>report: abstract and introduction</i>	1 hr	—	—	—
<i>report: data section</i>	2 hr	—	—	—
<i>report: machine learning section</i>	4 hr	—	—	—
<i>diagram state</i>	—	—	4 hr	—
<i>transitions implementation</i>	2 hr	—	7 hr	—
<i>user preference extraction</i>	3 hr	—	—	—
<i>restaurants retrieval</i>	—	3 hr	—	—
<i>report: dialog manager section</i>	1 hr	2 hr	—	1 hr
<i>extra requirements</i>	—	4 hr	—	1 hr
<i>inference rules</i>	—	—	—	2 hr
<i>report: reasoning section</i>	—	2 hr	—	—
<i>configurabilities</i>	3 hr	3 hr	—	—
<i>report: configurability section</i>	1 hr	—	—	—
<i>report: conclusion</i>	1 hr	—	—	—
<i>Subtotal (excl. group work)</i>	<b>24.5 hr</b>	<b>18.5 hr</b>	<b>14 hr</b>	<b>11 hr</b>
<i>group work</i>	6 hr	6 hr	6 hr	6 hr
<b>Total</b>	<b>30.5 hr</b>	<b>24.5 hr</b>	<b>20 hr</b>	<b>17 hr</b>

Table 6. Contributions table

In Table 6 we present the contribution of each group member to the several tasks achieved for developing the final project. In general, we tried to split the tasks between group members and then coordinate each other to make the different parts of the code work together properly.

The row "group work" in Table 6 refers to tasks that have been performed by everyone in the group, such as reading the assignment requirements, checking each other's code, and helping with debugging.

We retrospectively acknowledge that we did not manage to split tasks equally. This resulted in some group members contributing more than others, as it clearly appears in the table. Of course, we will learn from this and organize the work in a more balanced way among group members for Part 2.

## APPENDIX

## A: Full Dialog Flux Example

## Dialog 8

**system:** "Hello, welcome to B2's restaurant finder! Before starting to collect your preference about restaurants, we wish to ask you to select the configurability options that best suits you.  
 1: if you wish the system to always reply in CAPSLOCK, digit 'allcaps'  
 2: if you wish the system to delay its answers by 3 seconds, digit 'delay'  
 3: if you wish the system to classify your utterance based on a baseline instead of using a machine learning model, digit 'baseline'  
 4: if you wish the system to reply with an informal tone, digit 'informal'  
 You can choose multiple configurabilities at once, or also none of them."

**user:** "no config"

**system:** "Hello, welcome to B2's restaurant finder, please inform us of a pricerange, area and foodtype you are interested in."

**user:** "I want to eat at an expensive restaurant."

**system:** "Could you tell me your preferred area?"

**user:** "Center."

**system:** "Could you tell me your preferred food?"

**user:** "Any."

**system:** "We currently have found 23 restaurants that comply with your preferences. Would you like to give any additional requirements? We can filter on: child friendliness, romantic and touristic."

**user:** "I would like the restaurant to be romantic."

**system:** "We currently have found 5 restaurants that comply with your preferences. Would you like to give any additional requirements? We can filter on: child friendliness, romantic and touristic."

**user:** "I also want it child friendly."

**system:** "I'm sorry but the extra requirements you provided resulted in 0 matches, please provide us with new requirements or do not to get the restaurants we found without them."

**user:** "Okey, not child friendly, only romantic."

**system:** "We currently have found 5 restaurants that comply with your preferences. Would you like to give any additional requirements? We can filter on: child friendliness, romantic and touristic."

**user:** "Give me one."

**system:** "I found a nice restaurant called dojo noodle bar serving asian oriental food, at an expensive price in the centre part of town. The restaurant is romantic because people stay for long and it is not busy. Any other information about the restaurant we can help you with?"

**user:** "What is the address?"

**system:** "Sure — the addr is 40210 millers yard city centre."

**user:** "Could you also tell me the phone number?"

**system:** "Sure — the phone is 01223 363471."

**user:** "Okey, thank you very much."

**system:** "Thank you for using our service, any other information about the restaurant I can help you with?"

**user:** "No, bye bye."

**system:** "Thanks! Goodbye."

## B: Templates for 'informal' Configurability

State	Informal Template	Example
HELLO	"Hi welcom by B2 restaurant finder, give your pricerange, area and foodtype of choice"	"Hi welcom by B2 restaurant finder, give your pricerange, area and foodtype of choice"
PREFLOOP	"Tell me your preffered <slot>"	"Tell me your preffered area"
NOMATCH	"I couldn't find a <pricerange> restaurant serving <food> food in the <area> part of town. Do you have an alternative pricerange, area or food?"	"I couldn't find a cheap restaurant serving french food in the center part of town. Do you have an alternative pricerange, area or food?"
MATCHSUCCESS (no extra req.)	"I found <name> it has <food> for a <pricerange> in the <area> part of town. Do you need other information about the restaurant?"	"I found pizza hut fen ditton it has italian food for a moerate price in the east part of town. Do you need other information about the restaurant?"
MATCHSUCCESS (with extra req.)	"I found <name> it has <food> for a <pricerange> in the <area> part of town. It is <req> because <reason>. Do you need other information about the restaurant?"	"I found mahal of cambridge it has indian food for a cheap price in the centre part of town. It is touristic because it has good food and a cheap price. Do you need other information about the restaurant?"
USERMATCHED	"Thank you for using our service, do you need some other info?"	"Thank you for using our service, do you need some other info?"
FETCHDATA (error)	"Sorry something went wrong, we were not able to retrieve the data you asked for."	"Sorry something went wrong, we were not able to retrieve the data you asked for."
REQLOOP	"We found <n> restaurants that are okey with your preferences. Any other requirements? We can filter on: child friendliness, romantic and toursitic"	"We found 8 restaurants that are okey with your preferences. Any other requirements? We can filter on: child friendliness, romantic and toursitic"
NOREQMATCH	"The extra requirements you provided resulted in 0 matches, provide us with new requirements or do not to get the restaurants we found without them"	"The extra requirements you provided resulted in 0 matches, provide us with new requirements or do not to get the restaurants we found without them"
FETCHDATA (all requested fields found)	Same as formal template	Same as formal example
FETCHDATA (some found, some missing)	Same as formal template	Same as formal example
FETCHDATA (all missing)	Same as formal template	Same as formal example
EXIT	Same as formal template	Same as formal example

Table 7. System utterance templates and examples (informal style)

## REFERENCES

- [1] Suryakanthi Tangirala. 2020. Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm. *International Journal of Advanced Computer Science and Applications (IJACSA)* 11, 2 (2020).
- [2] Pejman Zarafshan, Saman Javadi, Abbas Roozbahani, Seyed Mehdi Hashemy, Payam Zarafshan, and Hamed Etezadi. [n. d.]. Artificial Intelligence Hybrid-Deep Learning Model for Groundwater Level Prediction Using MLP-ADAM. *Journal/Conference Name* ([n. d.]).